A Prototype of Exascale Checkpoint and Restart Computing Environment Using PLFS and Burst-Buffer Supports

Hsing-bung Chen, HPC-5; Gary A. Grider, HPC-DO; Parks M. Fields, Josip Loncaric, HPC-5; David R. Montoya, HPC-3

LANL has designed and developed a scalable and high performance Parallel Log-Structure File system (PLFS) for checkpoint and restart operations. PLFS has demonstrated its superior handling capability with parallel "N-to-1" file input/output (I/O) problems. Here we present a prototype Exascale checkpoint and restart computing environment using LANL's PLFS and local solid state device (SSD)-based burst-buffer storage enhancement supports. We use the global name space capability of the parallel file system as a metadata backend and the high-speed-data bandwidth of the local burst-buffer storage pool as a data backend. We illustrate a scalable checkpoint and restart system to meet Exascale computing requirements. We also propose three new collective MPI-IO application programming interfaces (API), MPI_File_protect_all, MPI_File_ensure_all, and MPI_File_delete_all, to support this prototype implementation, and use these to provide a space management sub-system between the burst-buffer and parallel file systems.

According to the technology roadmap of future Exascale computing systems, we expect checkpoint and restart file input/output (I/O) bandwidth to be in the range of 20 TB/sec to 60 TB/sec. Traditional disk-based parallel file systems cannot meet these challenging requirements in terms of cost, power consumption, reliability, and data access bandwidth. To meet these requirements for exascale computing, we need to introduce an additional layer to the traditional computer memory hierarchy. New storage-class memory technologies, such as solid state devices (SSD) and phase-change memory (PCM), provide very promising high-data bandwidth, high I/O operations per second (IOP) and low data-access latency.

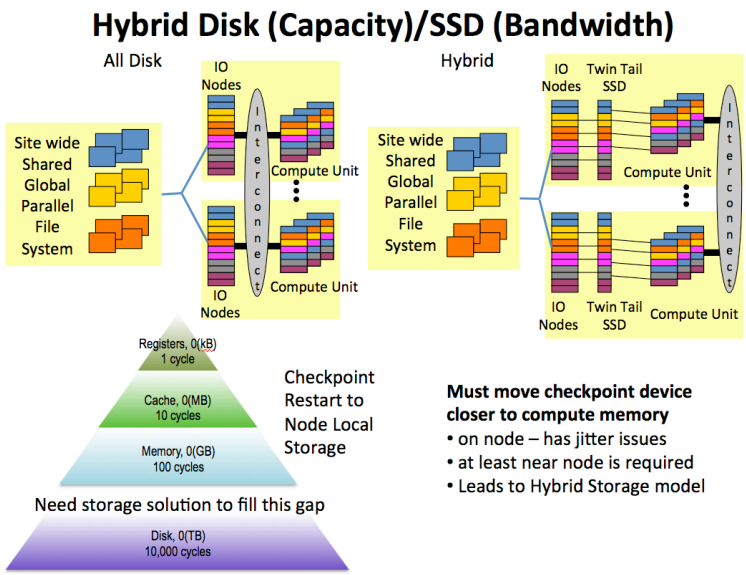

Fig. 1. A hybrid storage system: bridging capacity and bandwidth for exascale computing systems.

We intend to leverage these new storage technologies and use them as a localized read/write burst-buffer between memory and the traditional disk-based parallel file system. The combination of using a storage-class memory-based local burst buffer and disk-based parallel file system gives us a new prospect for the exascale checkpoint and restart computing environment. We first write checkpoint data to the local burst-buffer and later asynchronously migrate important long-term data from the burst buffer to a parallel file system. The burst buffer gives us memory-scale storage performance and, at the same time, disk-based parallel file systems can sustain the large storage demands from applications.

LANL has designed and developed a scalable and high-performance Parallel Log-Structure File system (PLFS) for checkpoint and restart file I/O operations. PLFS has already demonstrated its superior handling capability with parallel "N-to-1" file I/O problems. In this project, we present a prototype of the exascale checkpoint and restart computing environment using LANL's PLFS and local SSD-based burst-buffer storage enhancement supports. We utilize the global name space capability of the parallel file system as a metadata backend and the high-speed data bandwidth of the local burst-buffer storage pool as a data backend. We illustrate a scalable checkpoint and restart system to meet the exascale computing requirement. We also propose three new collective MPI-IO application programming interfaces (API): MPI_File_protect_all, MPI_File_ensure_all, and MPI_File_delete_all, to support this prototype implementation. We use these three new APIs to provide a space management sub-system between the burst-buffer and parallel file systems.

For more information contact Hsing-bung Chen at hbchen@lanl.gov

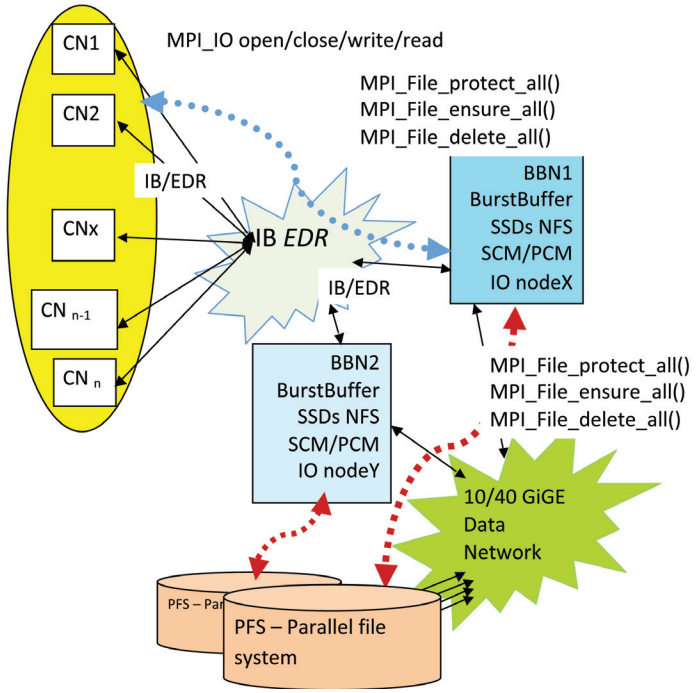

Fig. 2. A prototype of exascale checkpoint and restart system using hybrid storage system.

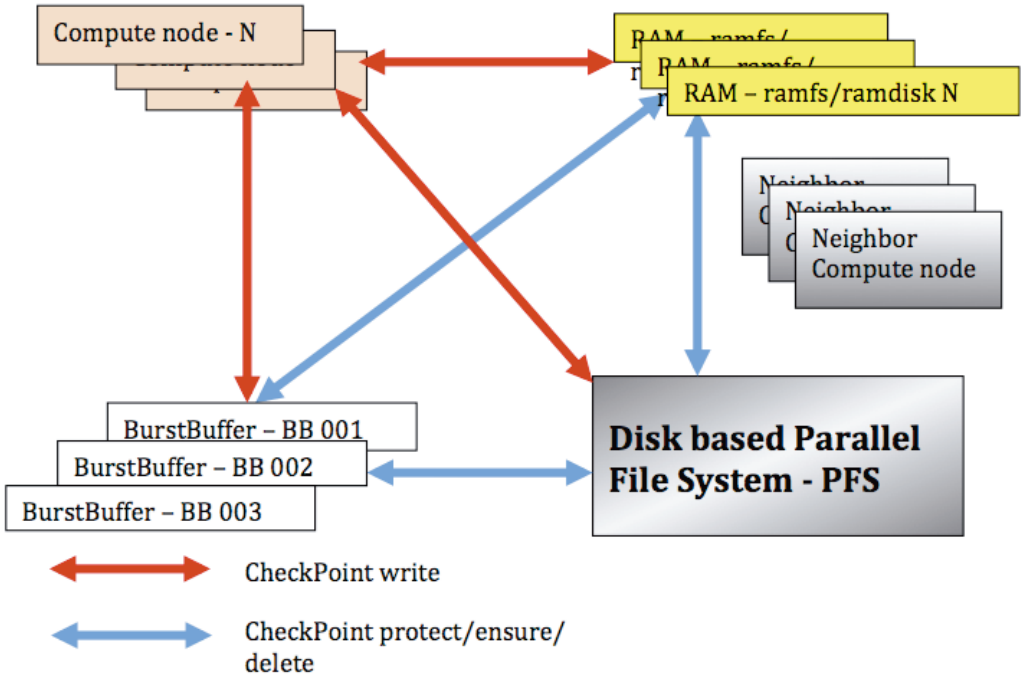

Fig. 3. Proposed three MPI-IO APIs to support burst-buffer checkpoint and restart operations.

Funding Acknowledgments
DOE NNSA, Advanced Simulation and Computing Program